Dossier d’analyse

# Extraire-Transformer-Charger

On commence l’analyse avec une exploration des données. A ce but, les fichiers sont lu utilisant la fonction de la librairie pandas « read\_csv() ». Il y a deux fichiers provenant de deux plateformes différentes : un fichier avec les *logs* sur lesquelles on veut baser notre modèle et nos prédictions, et un fichier *notes* avec les notes des apprenants qu’on veut prédire.

## Les données *logs*

Un fichier contentant les logs, qui représentent les traces d’activités de chaque apprenant d’un cours sur la plateforme ARCHE.

Les champs sont les suivants :

|  |  |  |  |
| --- | --- | --- | --- |
| **Champ** | **Description** | **Format** | **Spécifications** |
| heure | Horodatage de l’activité | Datetime  Année-mois-jour heure:minutes:secondes | Dates entre 23/02/2023 et 14/12/2024 |
| pseudo | Id de l’apprenant | Int représentant l’apprenant après anonymisation | Chiffre entre 5 et 1000 |
| contexte | Ressource objet de l’activité | Object (String: string) | 46 valeurs uniques |
| composant | Type d’activité | Object (string) | 17 valeurs uniques |
| evenement | Précisions sur l’activité | Object (string) | 61 valeurs uniques |

Une première analyse montre que :

* Le fichier contient 29 006 lignes et 5 colonnes
* Il n’y pas de valeurs manquantes

On constate qu’il y a des pseudos dans les logs qui n’ont pas de correspondance avec dans les notes. Les lignes avec ces pseudos sont enlevées.

La colonne « contexte » parait être un ensemble de deux éléments, séparés par un « : ». La colonne est découpée en deux colonnes : « contexte\_general » et « specification ». La colonne « contexte » est devenue obsolète et est alors supprimée.

Pour pouvoir travailler avec les notions jours et heures, on sépare également la colonne « heure » en deux partie, une colonne « jour » avec les dates et une colonne « heures » avec l’heure exacte. La première colonne est convertie en datetime, la deuxième en time. On garde la colonne « heure » pour faire des opérations avec pandas, qui ne reconnait pas le format « time ».

On obtient un dataframe qui :

* Contient 27 771 lignes et 8 colonnes

|  |  |  |  |
| --- | --- | --- | --- |
| **Champ** | **Description** | **Format** | **Spécifications** |
| heure | Horodatage de l’activité | Datetime  Année-mois-jour heure:minutes:secondes | Dates entre 23/02/2023 et 14/12/2024 |
| pseudo | Id de l’apprenant | Int représentant l’apprenant après anonymisation | Chiffre entre 5 et 1000 |
| composant | Type d’activité | Object (string) | 13 valeurs uniques |
| evenement | Précisions sur l’activité | Object (string) | 27 valeurs uniques |
| contexte\_general | Ressource objet de l’activité | Object (string) | 8 valeurs uniques |
| specification | Spécification sur la ressource | Object (string) | 45 valeurs uniques |
| jour | Date | Datetime  Année-mois-jour | Dates entre 23/02/2023 et 14/12/2024 |
| heures | Heure | Time (pas reconnu en pandas)  Heure:minutes:secondes | Dans pandas considéré comme object |

## Les données *notes*

Un fichier contentant les notes obtenues par les apprenants.

|  |  |  |  |
| --- | --- | --- | --- |
| **Champ** | **Description** | **Format** | **Description** |
| pseudo | Id de l’apprenant | Chiffre représentant l’apprenant après anonymisation | Forme le lien avec les données logs |
| note | Note obtenue par l’apprenant | String à cause de tirets (-) présents | Entre « - » et 19 |

Une première analyse montre que :

* Le fichier contient 80 lignes et 2 colonnes
* Il n’y pas de valeurs manquantes

Les notes sont interprétées comme des objets (string), à cause des tirets qui sont présents. Afin de les convertir en entiers, on remplace les tirets par des zéros.

Un histogramme de la distribution des notes montre que le taux de réussite est assez faible, avec seulement 30% des apprenants obtenant une note de 10 au plus. Les notes les plus fréquentes sont 2 et 3, suivis par 4 et 8. On suppose que les apprenants sont notés sur 20.

Pour le suivi des analyses les pseudos qui n’ont pas de correspondance dans les logs sont supprimés.

On obtient alors un dataframe avec 78 lignes et 2 colonnes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Champ** | **Description** | **Format** | **Description** |
| pseudo | Id de l’apprenant | Entier représentant l’apprenant après anonymisation | Forme le lien avec les données logs |
| note | Note obtenue par l’apprenant | Entier | Entre 0 et 19 |

On constate que le dataframe logs n’est pas en bon format pour le développement d’un modèle de machine learning. La construction d’un nouveau dataframe avec des indicateurs utilisables est alors nécessaire.

# Feature engineering

## Calcul des indicateurs

Les indicateurs (« features ») suivants sont calculés :

1. Le nombre d’actions réalisées par chaque apprenant
2. La moyenne du nombre d’actions réalisées par jour par chaque apprenant
3. Le nombre d’actions maximal réalisées par jour par chaque apprenant
4. La variabilité dans le nombre d’actions réalisées par jour par chaque apprenant
5. Le nombre de jours avec activité
6. La différence (en jours) entre la première et la dernière activité
7. La constance d’activité
8. Le pourcentage d’actions réalisées pendant le week-end
9. La période moyenne d’activité par jour
10. Le pourcentage d’actions réalisées pendant la nuit
11. Le pourcentage d’actions réalisées le matin
12. Le pourcentage d’actions réalisées l’après-midi
13. Le pourcentage d’actions réalisées le soir
14. Le nombre de composants utilisés par chaque apprenant
15. Le nombre d’événements utilisés par chaque apprenant
16. Le nombre de contextes générales utilisés par chaque apprenant
17. Le nombre d’actions pour chaque composant par apprenant
18. Le nombre d’actions pour chaque événement par apprenant
19. Le nombre d’actions pour chaque contexte général par apprenant
20. Le composant le plus utilisé par apprenant
21. L’événement le plus utilisé par apprenant
22. Le contexte général le plus utilisé par apprenant

On obtient alors un dataframe avec 78 lignes et 66 colonnes. Chaque ligne dans ce nouveau dataframe correspond à une ligne dans le dataframe des notes. Il n’y a pas de valeurs manquantes.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Distribution des features**

La plupart (54) des colonnes ont des entiers comme valeurs. Il y a 9 colonnes avec des floats et trois colonnes sont catégorielles. Les distributions des features sont diverses, mais ils ont tous au moins deux valeurs différentes.

**Relation features et target**

Il n’y a pas de relation significante entre les notes et les variables catégorielles.

Les corrélations entre les notes et les variables int et float varient entre -0,18 et 0,26. Il y a une corrélation positive (Pearson) significante entre les notes et :

* Evenement\_Fichier déposé
* Evenement\_Travail de devoir remis
* Composant\_Remises de fichiers
* Evenement\_Formulaire de remise consulté
* Evenement\_Travail de devoir modifié
* Evenement\_Travail de devoir créé

Les autres corrélations sont non-significatives.

**Relations feature x feature**

Il y a des très fortes corrélations entre les 6 features ayant une corrélation significative avec le target. Il y a également des corrélations fortes entre plusieurs autres features, ce qui n’est pas étonnante vue que les colonnes « contexte », « evenement » et « composant » sont intrinsèquement fortement liées et forme la base de plusieurs features. Nos analyses futures vont alors devoir prendre en compte cette multicollinearité.

Cependant, on constate plusieurs variables qui sont parfaitement corrélée l’un à l’autre (coefficient de corrélation égal à 1). Vue que ces variables contiennent exactement la même information, on va supprimer une colonne de chaque pair. Ainsi 14 variables sont supprimées.

## Feature transformation

**Train-test split**

Nous séparons le jeu de données en deux : une partie que nous allons utiliser pour l’entrainement du modèle (le trainset) et une partie que nous allons utiliser pour évaluer la performance de notre modèle obtenu (le testset). Nous choisissons une répartition de 80/20%, qui est le plus commun.

**Encoder**

Les 3 variables de type object sont catégorielles. Afin de pouvoir utiliser ces variables dans le développement d’un modèle de machine learning, on va devoir les transformer en valeurs numériques, c’est-à-dire, les encoder. On a des variables catégorielle (sans ordre significatif) avec plusieurs catégories, donc on utilise la méthode OneHot pour encoder les catégories dans plusieurs colonnes appelé dummies.

Le dataframe contient maintenant que des valeurs numériques.

**Normalisation des variables**

## Feature selection

# Modélisation

## Régression

## Autre modèle

# Evaluation

## Métriques utilisées

## Performances